Q1:CRC

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N strlen(g)

char t[128],cs[128],g[]="10001000000100001";
int a,e,c;

void xor()
{
   for(c=1;c<N;c++)
   {
      cs[c]=(cs[c]==g[c])?'0':'1';
   }
}
void crc()
{
   for(e=0;e<N;e++)
   {
      cs[e]=t[e];
   }
   do
   {
   if(cs[0]=='1')
   xor();
   for(c=0;c<N-1;c++)
   {
      cs[c]=cs[c+1];
   }
   cs[c]=t[e++];
   }while(e<=a+N-1);
}
void main()
{
   printf("enter the polynomial\n");
   scanf("%s",t);
   printf("generating polynomial is: %s\n",g);
   a=strlen(t);
   for(e=a;e<a+N-1;e++)
   {
      t[e]='0';
   }
   printf("modified t[u] is %s\n",t);
   crc();
```

```
    printf("check sum: %s\n",cs);
    for(e=a;e<a+N-1;e++)
        t[e]=cs[e-a];
        printf("final code word is: %s\n",t);
        printf("test error 0 yes 1 no\n");
        scanf("%d",&e);
        if(e==0)
        {
            printf("enter position to introduce error\n");
            scanf("%d",&e);
            t[e]=(t[e]=='0')?'1':'0';
            printf("erreneous data: %s\n",t);
        }
        crc();
        for(e=0;(e<N-1&&cs[e]!='1');e++);
        if(e<N-1)
            printf("error detected\n");
        else
            printf("error not detected\n");
}
```

Q2: RSA

```
#include <stdio.h>
#include <stdlib.h>
int n,z,i,j,e,d,p,q,d,ct[20],pt[20];
char msg[20];
int gcd(int x,int y)
{
    while(x!=y)
    {
        if(x>y)
            x=x-y;
        else
            y=y-x;
    }
    return x;
}
int main()
{
    p=11;
    q=17;
    n=p*q;
    z=(p-1)*(q-1);
    printf("enter the message\n");
    gets(msg);
```

```
  e=1;
    do
    {
        e++;
    }while((gcd(e,z)!=1)&&(e<z));
    for(d=1;d<z;d++)
    {
        if((e*d)%z==1)
            break;
    }
    printf("n:%d\t z:%d\t e:%d\t d:%d\t",n,z,e,d);
    printf("\nencryption\n");

    for(j=0;j<strlen(msg);j++)
    {
        ct[j]=1;
        for(i=0;i<e;i++)
        ct[j]=(ct[j]*msg[j])%n;
    }
    for(i=0;i<strlen(msg);i++)
    {
        printf("%d\t",ct[i]);
    }
    printf("\n");
    printf("\ndecryption\n");
    for(j=0;j<strlen(msg);j++)
    {
        pt[j]=1;
        for(i=0;i<d;i++)
        pt[j]=(pt[j]*ct[j])%n;
    }
    for(i=0;i<strlen(msg);i++)
    {
        printf("%c",pt[i]);
    }
    printf("\t");
    return 0;
}
```

Q3:HAMCODE

```
#include <stdio.h>
#include <stdlib.h>
int h[11];
void correct_error();
void generate_hammcode()
```

```
{
    int temp,i;
    temp=h[3]+h[5]+h[7]+h[9]+h[11];
    (temp%2!=0)?(h[1]=1):(h[1]=0);
     temp=h[3]+h[6]+h[7]+h[10]+h[11];
    (temp%2!=0)?(h[2]=1):(h[2]=0);
     temp=h[5]+h[6]+h[7];
    (temp%2!=0)?(h[4]=1):(h[4]=0);
     temp=h[9]+h[10]+h[11];
    (temp%2!=0)?(h[8]=1):(h[8]=0);
    for(i=11;i>=1;i--)
    {
        printf("%d\t",h[i]);
    }
    printf("\n");
}
void make_error()
{
    int pos,i;
    printf("enter position to make error\n");
    scanf("%d",&pos);
    if(h[pos]==0)
        h[pos]=1;
    else
        h[pos]=0;
    printf("erroneous data\n");
    for(i=11;i>=1;i--)
    {
        printf("%d\t",h[i]);
    }
    printf("\n");
    correct_error();
}
void correct_error()
{
    int p1,p2,p4,p8,errpos,i;
     p1=(h[1]+h[3]+h[5]+h[7]+h[9]+h[11])%2;
     p2=(h[2]+h[3]+h[6]+h[7]+h[10]+h[11])%2;
     p4=(h[4]+h[5]+h[6]+h[7])%2;
     p8=(h[8]+h[9]+h[10]+h[11])%2;
     errpos=p8*8+p4*4+p2*2+p1*1;
     if(h[errpos]==1)
        h[errpos]=0;
     else
        h[errpos]=1;
     printf("error detected at %d\n",errpos);
     printf("error corrected\n");
     for(i=11;i>=1;i--)
    {
       printf("%d\t",h[i]);
    }
    printf("\n");
}
int main()
{
    int i,error;
    printf("enter data in bits\n");
    for(i=11;i>=1;i--)
    {
       if(i==1||i==2||i==4||i==8)
           continue;
       scanf("%d",&h[i]);
    }
    generate_hammcode();
    printf("enter 1(yes)/0(no) to enter and correct error\n");
    scanf("%d",&error);
    if(error==1)
       make_error();
    else
       printf("received without error\\n");
    return 0;
}
```

Q4:FRAME SORT

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#define DATA_SZ 3
typedef struct packet
{
int SeqNum;
char Data[DATA_SZ+1];
}
PACKET;
PACKET *readdata, *transdata;
time_t t ;
int divide(char *msg)
{
int msglen, NoOfPacket, i, j;
```

```
msglen = strlen(msg);
NoOfPacket = msglen/DATA_SZ;
if( (msglen%DATA_SZ) !=0)
NoOfPacket++;
readdata = (PACKET *)malloc(sizeof(PACKET) *NoOfPacket);
for(i = 0; i < NoOfPacket; i++)
{
readdata[i].SeqNum = i + 1;
for (j = 0;(j < DATA_SZ) && (*msg != '\0'); j++, msg++)
readdata[i].Data[j] = *msg;
readdata[i].Data[j] = '\0';
}
printf("\nThe Message has been divided as follows\n");
printf("\nPacket No.\tData\n\n");
for (i = 0; i < NoOfPacket; i++)
printf(" %d\t\t%s\n", readdata[i].SeqNum, readdata[i].Data);
return NoOfPacket;
}
void shuffle(int NoOfPacket)
{
int *Status;
int i, j, trans;
srand(time(&t));
Status=(int * )calloc(NoOfPacket, sizeof(int));
transdata = (PACKET *)malloc(sizeof(PACKET) * NoOfPacket);
for (i = 0; i < NoOfPacket;)
{
trans = rand()%NoOfPacket;
if (Status[trans]!=1)
{
transdata[i].SeqNum = readdata[trans].SeqNum;

strcpy(transdata[i].Data, readdata[trans].Data);
i++;
Status[trans] = 1;
}
}
free(Status);
}
void sortframes(int NoOfPacket)
{
PACKET temp;
int i,j;
for (i = 0; i < NoOfPacket; i++)
{
```

```
for (j = 0;j < NoOfPacket - (i+1); j++)
{
if (transdata[j].SeqNum > transdata[j + 1].SeqNum)
{
temp.SeqNum = transdata[j].SeqNum;
strcpy(temp.Data, transdata[j].Data);
transdata[j].SeqNum = transdata[j + 1].SeqNum;
strcpy(transdata[j].Data, transdata[j + 1].Data);
transdata[j + 1].SeqNum = temp.SeqNum;
strcpy(transdata[j + 1].Data, temp.Data);
}
}
}
}
void receive(int NoOfPacket)
{
int i;
printf("\nPackets received in the following order\n");
for (i = 0; i < NoOfPacket; i++)
printf("%4d",transdata[i].SeqNum);
sortframes(NoOfPacket);
printf("\n\nPackets in order after sorting..\n");
for (i = 0; i < NoOfPacket; i++)
printf("%4d",transdata[i].SeqNum);
printf("\n\nMessage received is :\n");
for (i = 0; i < NoOfPacket; i++)
printf("%s",transdata[i].Data);
}
int main ()
{
char msg[25];
int NoOfPacket;
printf("\nEnter The message to be Transmitted :\n");
scanf("%[^\n]", msg);
NoOfPacket = divide(msg);
shuffle(NoOfPacket);
receive(NoOfPacket);
free(readdata);
free(transdata);
return 0;
}
```

Q5:DISTANCE VECTOR

```
 #include<stdio.h>
#define INFINITY 99
```

```
struct
{
        int cost;
        int via;
}
routeTable[10][10];
int n;
int main()
{
int src,dst,i;
Int opt;
printf("Enter the Number of routers:");
scanf("%d",&n);
read_route_table();
for(i=0;i<n;i++)
build_route_table(i);
for(i=0;i<n;i++)
disp_route_table(i);
do
{
printf("\nEnter the Source node(0 to %d): ",n-1);
scanf("%d",&src);
printf("Enter the Destination node(0 to %d):",n-1);
scanf("%d",&dst);
if(src > (n - 1) || dst > (n - 1))
printf("\n router doest not exist");
else
{
find_path(src,dst);
printf("\nThe cost of the shortest route is:\t%d\n",routeTable[src][dst].cost);
}
printf("\nDo you want to continue? (0/1):\n");
scanf("%d",&opt);
}
while(opt);
return 0;
}
void read_route_table()
{
int i,j;
printf("Enter the initial routing table (if no direct link, enter 99):\n");
for(i=0;i<n;i++)
{
printf("\nRouting table for %c:\n",'A' + i);
for(j=0;j<n;j++)
{
if(i==j)
routeTable[i][j].cost=0;
else
{
printf("--> %c:",'A' + j);
scanf("%d",&routeTable[i][j].cost);
}
if(routeTable[i][j].cost != INFINITY)
routeTable[i][j].via=j;
else
routeTable[i][j].via=INFINITY;
}
}
}
void build_route_table(int x)
{
int i,j,new_cost=0;
for(i=0;i<n;i++)
{
for(j=0;j<n && i!=x;j++)
{
if(routeTable[x][i].cost != INFINITY)
{
new_cost =routeTable[x][i].cost+routeTable[i][j].cost;
if(routeTable[x][j].cost > new_cost)
{
routeTable[x][j].cost=new_cost;
routeTable[x][j].via=routeTable[x][i].via;
}}}}}
void disp_route_table(int x)
{
int i;
printf("\nFinal Routing Table for %c: ",'A' + x);
printf("\n\tDestination\tCost\tOutgoing line");
printf("\n\t-------------\t----\t--------------\n");
for(i=0;i<n;i++)
{
printf("\n\t\t%c",'A' + i);
printf("\t%d",routeTable[x][i].cost);
printf("\t%c",'A' + routeTable[x][i].via);
printf("\n");
}
}
void find_path(int x,int y)
```

```
{
printf("%c",'A' + x);
if(x != y)
{
printf(" --> ");
find_path(routeTable[x][y].via,y);
}
}
```

Q6:LEAKY BUCKET

```
#include<stdio.h>
#include<stdlib.h>
//#define BUCKETSIZE 25
//#define OUTRATE 10
struct pkt
{
        int arrtime;
        int weight;
}
p[10];
int main()
{
        int BUCKETSIZE,OUTRATE;
        printf("Enter the bucket size\n");
        scanf("%d",&BUCKETSIZE);
        printf("Enter the out rate\n");
        scanf("%d",&OUTRATE);
        int i,n,excess=BUCKETSIZE;
        int j=0,rem=0;
        printf("Enter the number of packets");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
        printf("Enter arrival time\t");
        scanf("%d",&p[i].arrtime);
        //p[i].arrtime=rand();
        p[i].weight=rand()%BUCKETSIZE;
        printf("packet size id %d\n",p[i].weight);
        //scanf("%d",&p[i].weight);
        }
        for(i=0;i<=40;i++)
        {
        if(p[j].arrtime==i)
        {
        if(p[j].weight<=excess)
        {
        rem=p[j].weight+rem;
        excess=excess-p[j].weight;
        printf("At time=%d: packet %d inserted into bucket,",i,j+1);
        printf(" remaining bucket size= %d\n",excess);
        j=j+1;
        }
         else
        {
        printf("At time = %d: packet %d discarded,",i,j+1);
        printf("Packet size is more than buffer size\n");
        j=j+1;
         } }
        if((i%5)==0)
        {
        if(rem>=OUTRATE)
        {
        rem=rem-OUTRATE;
        excess=excess+OUTRATE;
        printf("At time = %d : 10 Kbytes are transfered ",i);
        printf("Free available space in the bucket=%d\n",excess);
        }
        else if(rem>0)
        {
        excess=excess+rem;
        printf("At time = %d : %d Kbytes are transfered ",i,rem);
        printf("Free available space in the bucket=%d\n",excess);
        rem=0;
        }
        }
        }
printf("Bucket is empty");
}
```

Q7:CLIENT

```
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
int main(int argc,char *argv[])
{
        int create_socket,cont;
        int bufsize=1024;
```

```
	char *buffer=malloc(bufsize);
	char fname[256];
	struct sockaddr_in address;
	if((create_socket=socket(AF_INET,SOCK_STREAM,0))>0)
	printf("The socket was created\n");
	address.sin_family=AF_INET;
	address.sin_port=htons(15000);
	inet_pton(AF_INET,argv[1],&address.sin_addr);
	if(connect(create_socket,(struct sockaddr*)&address,sizeof(address))==0)
	printf("The connection was accepted with server %s...\n",argv[1]);
	printf("Enter the filename to request:");
	scanf("%s",fname);
	send(create_socket,fname,sizeof(fname),0);
	printf("Request Accepted... Receiving File... \n\n");
	printf("The contents of file are...\n\n");
	while((cont=recv(create_socket,buffer,bufsize,0))>0)
	{
		write(1,buffer,cont);
	}
	printf("\nEOF\n");
	return close(create_socket);
}
```

SERVER:

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<fcntl.h>
int main()
{
int cont,create_socket,new_socket,addrlen,fd;
int bufsize=1024;
char *buffer=malloc(bufsize);
char fname[256];
struct sockaddr_in address;
if((create_socket=socket(AF_INET,SOCK_STREAM,0))>0)
{	printf("The socket was created\n");
}
address.sin_family=AF_INET;
address.sin_addr.s_addr=INADDR_ANY;
address.sin_port=htons(15000);
if(bind(create_socket,(struct sockaddr *)&address,sizeof(address))==0)
{       printf("Binding Socket\n");      }
listen(create_socket,3);
addrlen=sizeof(struct sockaddr_in);
new_socket=accept(create_socket,(struct sockaddr *)&address,&addrlen);
if(new_socket>0)
{       printf("The Client is connected...\n"); }
recv(new_socket,fname,255,0);
printf("A request for filename %s Received...\n",fname);
if((fd=open(fname,O_RDONLY))<0)
{perror("File Open Failed");
exit(0); }
while((cont=read(fd,buffer,bufsize))>0)
{ send(new_socket,buffer,cont,0); }
printf("Request Completed\n");
close(new_socket);
return close(create_socket);
}
```